LINUX SHELL

An Interactive Shell Made Using Python

Project Abstract

The project is a custom Linux shell which incorporates file handling commands, process handling commands and some custom commands and features for the shell.

A shell, although not part of the operating system, is the primary interface between the user and the operating system. It gathers inputs from a user (command line interpreter), executes programs based on the input by making system calls to the kernel, and displays the output.

The shell is the outermost layer of the operating system, it incorporates a programming language to control processes and files, as well as to start and control other programs.

Our version of a shell contains a few custom commands that we find necessary to a shell and the implementation of a few basic (pre-existing) commands of the Linux shell. All of the commands that we have implemented are woven around a central theme that our shell will portray. Through this project, we hope to demonstrate concepts like File Handling, Process Management and interaction of file systems with processes, System Calls and a few others.

The basic Linux terminal commands that we have implemented in our shell are:

1. cat - helps is view contents of a specific file, or create/write to a file, or append contents of one file to another
2. mkdir/rmdir - deals with the management of directories in a file system for creating a new directory or removing an existing directory
3. file - utilises lookups on extensions and file headers to determine the type and metadata about the file
4. free - gives information about memory usage in the system
5. pwd - locates the current working directory of the user
6. help/man - presents information about every command available in the shell

The other complex and custom commands we have worked on are:

1. File Searcher - searches for a particular file in a directory specified and returns the file path, for further operations on the file
2. Running Processes - displays information related to processes that are running on the system. View all PIDs of the processes, view process status information by specifying process id, view the memory usage of any running process.

Our shell also incorporates the fuzzy autocomplete feature → It completes any command for the user, irrespective of any minor errors that might occur in the command by performing a lookup of the recently used commands.

Programming Language Used

We have used Python as the programming language to write scripts that would run our custom shell. With the added advantage and flexibility of using in-built functions and external modules, this was a convenient use case for Python programming. The whole shell program will be written using the Python 3.0 programming language and will utilise certain external libraries like *magic* and *psutil* which simplifies the process of determining file and process information while retaining the same functionality.

Code

The commands have been named according to the nature of the commands. The implementation of our commands are based on, but not exactly the same as, the following existing commands:

- man --> treasure
- help --> sos
- cat --> glance
- file --> fish
- free --> engine
- mkdir --> forge
- ps --> radar
- rename --> rename
- rmdir --> dismantle
- searcher --> dive

Codes for all the commands and other used codes are available on the zip file.

Learning Outcome

After the successful execution of the ideas presented in this synopsis, we as a team will become familiar with the design of a shell and how a user interacts with the service and programs of an operating system.

Through implementation, our team members learnt to utilise various system calls and gained a better understanding of core operating system ideas like processes, memory, file systems, directories etc. The shell exposes you to the filesystem more directly than the graphical file browser and makes you understand the hierarchy and structure of the OS. You also get to play with the configuration files directly and this gives you the power to control your operating system more efficiently.

The expected output of the project will be a fully functional custom shell with the above-mentioned commands and features. A help page for the whole shell will be added for user comfort, with suitable prompts wherever required.

The concepts learnt from the individual commands are:
→ monitoring processes, and the files opened by processes: ps
→ check and monitor memory usage: free
→ file handling and management: cat, file, file-searcher
→ handling directories and files and learning the hierarchy of the file system: rmdir, mkdir

Lastly, by the end of the project, we would have developed our own shell with several useful features and in the process would have gained invaluable knowledge about programming and development through implementation.

Future Scope

The straight answer is, “No, there is no big scope only for Linux Shell only”. Linux shell provides you with an interface to the Unix system. It gathers input from you and executes programs based on that input. When a program finishes executing, it displays that program's output. Shell is an environment in which we can run our commands, programs, and shell scripts. Shell alone doesn’t have a future scope but when combined with various programs, scrips and commands it is able to bring out the best of the present technology. In the future a lot of the things are going to be automated and shell can help in the automation.

References

1. psutil documentation — psutil 5.9.0 documentation
2. https://www.journaldev.com/43930/process-management-in-linux 3. https://realpython.com/working-with-files-in-python/
4. https://linuxcommand.org/lc3_learning_the_shell.php